EXAMINER'S AMENDMENT

1. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

2. Authorization for this examiner's amendment was given in a telephone interview with Kuiran Liu, Reg. No. 60,039 on 9/22/2010.

3. The claims have been amended as follows:

1-33. (Canceled)

34. (Currently Amended) A system for supporting resource enlistment synchronization, comprising:

an application server with a plurality of threads, wherein the application server runs on one or more processors and is associated with a plurality of applications, wherein each application is associated with at least one said thread;

at least one resource object, wherein the at least one resource object is associated with a resource connection object;

a transaction manager that manages a plurality of transactions, wherein each transaction is associated with ~~a said~~ each application;

wherein the resource connection object operates to perform:

receiving a request, to access the at least one resource object that is associated with the resource connection object, from a an application of the plurality of applications that runs on a thread of the plurality of threads, and

placing a call to the transaction manager and informing the transaction manager that current work performed by the at least one resource object is to be associated with a current transaction that is associated with the application, and

wherein, after receiving the call from the resource connection object, the transaction manager operates to perform:

first checking to see if there is an in-progress enlistment of the at least one resource object by another thread in another transaction,

if there is a lock,

blocking the request to enlist the resource object in the transaction and preventing different transactions from enlisting a logical connection to the at least one resource object at same time, and

initiating the at least one resource object to perform work associated with the thread and the current transaction, after the at least one resource object is delisted from another transaction that owns the lock,

if there is no lock,

enlisting the at least one resource object in the transaction and signaling the at least one resource object to begin processing the request,

receiving from the at least one resource object a delist resource method call, after a result is obtained for the request, to delist the at least one resource object from the current transaction and provides the result to the ~~first said~~ application, and

initiating the at least one resource object to perform work associated with another thread and another transaction.

35. (Canceled).

36. (New) The system of Claim 34, wherein:

the resource connection object is periodically processed to remove objects that are unused or no longer active.

37. (New) The system of Claim 34, wherein:

the at least one resource object resides in a server node.

38. (New) The system of Claim 34, wherein:

the transaction manager uses a priority method to determine which thread will be granted a lock.

39. (New) The system of Claim 34, wherein:

after the thread obtains a lock, the thread uses the resource connection object to initiate work on the at least one resource object.

40. (New) The system of Claim 34, wherein:

the resource connection object sends a delist call to the transaction manager and the transaction manager sends an end call to the at least one resource object to end work performed by the at least one resource object associated with the thread and release the lock on the at least one resource object.

41. (New) The system of Claim 34, wherein:

once the transaction manager enlists the resource object and obtains a lock to the at least one resource object, any attempted enlist from a second thread is blocked.

42. (New) The method of Claim 34, wherein:

the transaction manager operates to

determine whether an application associated with the thread is a specific type of application; and

grant the thread a lock only when the application is determined to be the specific type of application.

43. (New) A method for supporting resource enlistment synchronization, comprising:

associating a plurality of applications with an application server with a plurality of threads, wherein the application server runs on one or more processors, and wherein each application is associated with at least one said thread;

providing at least one resource object, wherein the at least one resource object is associated with a resource connection object;

providing a transaction manager that manages a plurality of transactions, wherein each transaction is associated with each application;

allowing the resource connection object to perform:

receiving a request, to access the at least one resource object that is associated with the resource connection object, from an application of the plurality of applications that runs on a thread of the plurality of threads, and

placing a call to the transaction manager and informing the transaction manager that current work performed by the at least one resource object is to be associated with a current transaction that is associated with the application; and

after receiving the call from the resource connection object, allowing the transaction manager to perform:

first checking to see if there is an in-progress enlistment of the at least one resource object by another thread in another transaction,

if there is a lock,

blocking the request to enlist the resource object in the transaction and preventing different transactions from enlisting a logical connection to the at least one resource object at same time, and

initiating the at least one resource object to perform work associated with the thread and the current transaction, after the at least one resource object is delisted from another transaction that owns the lock,

if there is no lock,

enlisting the at least one resource object in the transaction and signaling the at least one resource object to begin processing the request,

receiving from the at least one resource object a delist resource method call, after a result is obtained for the request, to delist the at least one resource object from the current transaction and provides the result to the application, and

initiating the at least one resource object to perform work associated with another thread and another transaction.

44. (New) The method of Claim 43, further comprising:

periodically processing the resource connection object to remove objects that are unused or no longer active.

45. (New) The method of Claim 43, further comprising:

allowing the at least one resource object to reside in a server node.

46. (New) The method of Claim 43, further comprising:

allowing the transaction manager to use a priority method to determine which thread will be granted a lock.

47. (New) The method of Claim 43, further comprising:

after the thread obtains a lock, allowing the thread to use the resource connection object to initiate work on the at least one resource object.

48. (New) The method of Claim 43, further comprising:

sending, via the resource connection object, a delist call to the transaction manager, and

sending, via the transaction manager, an end call to the at least one resource object to end work performed by the at least one resource object associated with the thread and release the lock on the at least one resource object.

49. (New) The method of Claim 43, further comprising:

blocking any attempted enlist from a second thread, once the transaction manager enlists the at least one resource object and obtains a lock to the at least one resource object.

50. (New) The method of Claim 43, further comprising:

determining, via the transaction manager, whether an application associated with the thread is a specific type of application; and

granting, via the transaction manager, the thread a lock only when the application is determined to be the specific type of application.

51. (New) A computer-readable storage medium, storing instructions for supporting resource enlistment synchronization, the instructions comprising the steps of:

associating a plurality of applications with an application server with a plurality of threads, wherein the application server runs on one or more

processors, and wherein each application is associated with at least one said thread;

providing at least one resource object, wherein the at least one resource object is associated with a resource connection object;

providing a transaction manager that manages a plurality of transactions, wherein each transaction is associated with each application;

allowing the resource connection object to perform:

receiving a request, to access the at least one resource object that is associated with the resource connection object, from an application of the plurality of applications that runs on a thread of the plurality of threads, and

placing a call to the transaction manager and informing the transaction manager that current work performed by the at least one resource object is to be associated with a current transaction that is associated with the application; and

after receiving the call from the resource connection object, allowing the transaction manager to perform:

first checking to see if there is an in-progress enlistment of the at least one resource object by another thread in another transaction,

if there is a lock,

blocking the request to enlist the resource object in the transaction and preventing different transactions from enlisting a

logical connection to the at least one resource object at same time, and

initiating the at least one resource object to perform work associated with the thread and the current transaction, after the at least one resource object is delisted from another transaction that owns the lock,

if there is no lock,

enlisting the at least one resource object in the transaction and signaling the at least one resource object to begin processing the request,

receiving from the at least one resource object a delist resource method call, after a result is obtained for the request, to delist the at least one resource object from the current transaction and provides the result to the application, and

initiating the at least one resource object to perform work associated with another thread and another transaction.

REASONS FOR ALLOWANCE

Claims 34, 36-51 are allowed and renumbered as 1-17

Claims 1-33 and 35 have been canceled.

The following is an Examiner's statement of reasons for the indication of allowable subject matter: Claims 34, 36-51 are allowable over the prior art of records because the Examiner found neither prior art cited in its entirely, nor based on the prior art, found any motivation to combine any of the prior arts.

The reason for allowance for claims 34, 43 and 51 is receiving a request, to access the at least one resource object that is associated with the resource connection object, from an application of the plurality of applications that runs on a thread of the plurality of threads, and placing a call to the transaction manager and informing the transaction manager that current work performed by the at least one resource object is to be associated with a current transaction that is associated with the application; and after receiving the call from the resource connection object, allowing the transaction manager to perform: first checking to see if there is an in-progress enlistment of the at least one resource object by another thread in another transaction, if there is a lock, blocking the request to enlist the resource object in the transaction and preventing different transactions from enlisting a logical connection to the at least one resource object at same time, and initiating the at least one resource object to perform work associated with the thread and the current transaction, after the at least one resource object is delisted from another transaction that owns the lock, if there is no lock, enlisting the at least one resource object in the transaction and signaling the at least one resource object to begin processing the request, receiving from the at

least one resource object a delist resource method call, after a result is obtained for the request, to delist the at least one resource object from the current transaction and provides the result to the application, and initiating the at least one resource object to perform work associated with another thread and another transaction, along with other features, as cited in the independent claims 34, 43 and 51.

4. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Camquy Truong whose telephone number is (571)272-3773. The examiner can normally be reached on 9:00am - 5:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Emerson C. Puente can be reached on (571)272-3645. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Camquy Truong/
Examiner, Art Unit 2195

/Emerson C Puente/
Supervisory Patent Examiner, Art Unit 2195